ADVENTIST UNIVERSITY OF CENTRAL AFRICA

**Name:** Nzamwitakuze Fabrice 08/10/2023

**ID:** 24855

**Course Name:** Programming with Java

**ASSIGNMENT1**

1. **The list of some popular types of NoSQL databases:**
   A. **MongoDB:** MongoDB is a popular document-oriented NoSQL database that stores data in BSON (Binary JSON) format. It is known for its flexibility and scalability, making it suitable for a wide range of applications**.**
   B. **Redis**: Redis is a high-performance, in-memory key-value store. It is commonly used for caching, real-time analytics, and other applications that require low-latency data access.
   C. **Apache Cassandra:** Cassandra is a distributed column-family NoSQL database designed for scalability and high availability. It is commonly used in applications where fast writes and high availability are critical, such as IoT and time-series data.
   D. **Neo4j:** Neo4j is a popular graph database that is designed to store and query data in the form of nodes and relationships. It is well-suited for applications that require complex graph-based queries, such as social networks and recommendation engines.
   E. **Apache HBase:** HBase is a distributed wide-column store that is often used for storing large amounts of sparse data. It is modeled after Google Bigtable and is commonly used in Hadoop-based ecosystems.
   F. **InfluxDB:** InfluxDB is a time-series database that is optimized for storing and querying time-stamped data. It is commonly used in applications like monitoring, IoT, and real-time analytics.

**Conclusion:**

The best NoSQL database depends on the specific use case, requirements, and trade-offs you are willing to make. Factors to consider when choosing a NoSQL

database include scalability, data modeling needs, consistency requirements, query complexity, and operational ease.

2. **The behavior of execute() Method in java**

    - **The execute() method in java** returns true if the SQL query it executes returns a ResultSet, typically associated with SELECT queries. This indicates that the query executed successfully and produced a result set.
    - **For non-SELECT queries and DDL queries, execute() method returns false** because these queries don't return result sets but rather indicate the number of affected rows.

3. **The difference between arguments and parameters in java**
   **Parameters:**

   **Parameters**
    - are the variables declared in a method or constructor definition.
    - They act as placeholders for values that will be passed into the method when it's called.
    - Parameters are defined in the method signature and specify the type and name of the values that the method expects.
      Example:

```
public void calculateSum(int num1, int num2) {
   // num1 and num2 are parameters
   int sum = num1 + num2;
   System.out.println("Sum: " + sum);
}
```

   **Arguments:**

    - Arguments are the actual values or expressions that you provide when calling a method or constructor.
    - They are the values that are passed into the method and assigned to the corresponding parameters.

Example:
int a = 5;
int b = 3;
calculateSum(a, b); **// Here, a and b are arguments**

4. **The Id validation In java**

```
System.out.println("Enter book ID with 3 characters");
bookId = sc.next();
if(bookId.length()==3){
    System.out.println("Enter boo Tile");
bookTitle = sc.next();
// create model object

theBook.setBookId(bookId);
theBook.setTitle(bookTitle);
// create dao object

String feedBack =  dao.insertBook(theBook);
System.out.println(feedBack);

System.out.println("Enter yes or no to continue or to quit the prc
String answer = sc.next();

if(answer.equalsIgnoreCase("yes")){
    condition = true;
}else{
     System.out.println("Thank you for using the system");
    condition = false;
}
}else{
    System.out.println("Invalid Id");
}
```